In the Claims:

Please amend claims 71-90, as indicated below.

1. (Original) A method for representing computer programming language objects in a data representation language, the method comprising:

a process executing within a virtual machine providing a first computer programming language object to a compilation process of the virtual machine, wherein the first object is an instance of a class in the computer programming language; and

the compilation process of the virtual machine converting the first object into a data representation language representation of the first object;

wherein the data representation language representation of the first object is configured for use in generating a copy of the first object.

2. (Original) The method as recited in claim 1, wherein the first object references one or more computer programming language objects, and wherein the compilation process of the virtual machine converting the first object into a data representation language representation of the first object comprises the compilation process converting the one or more objects into data representation language representations of the one or more objects.

3. (Original) The method as recited in claim 1, wherein the compilation process of the virtual machine converting the first object into a data representation language representation of the first object comprises:

processing the first object into an intermediary table representation of the first object; and

processing the intermediary table representation of the first object into the data representation language representation of the first object.

4. (Original) The method as recited in claim 3, wherein the first object comprises one or more instance variables, and wherein said processing the first object into an intermediary table representation comprises:

for each of the one or more instance variables in the first object, generating an entry in the intermediary table representation of the first object, wherein the entry for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable.

5. (Original) The method as recited in claim 4, wherein the first object comprises a plurality of instance variables with the same identifier, and wherein the entry for each of the plurality of instance variables with the same identifier further includes an enumeration value that uniquely identifies the instance variable in the plurality of instance variables with the same identifier.

6. (Original) The method as recited in claim 4, wherein said processing the intermediary table representation of the first object into the data representation language representation of the first object comprises:

for each of one or more entries in the intermediary table representation of the first object, generating a corresponding element in the data representation language representation of the first object, wherein the element in the data representation language representation of the first object includes an identifier of the instance variable and a value of the instance variable.

7. (Original) The method as recited in claim 6, wherein the one or more elements in the data representation language representation of the first object are configured for use in initializing one or more corresponding instance variables in the copy of the first object.

8. (Original) The method as recited in claim 1, further comprising providing an application programming interface (API) for the compilation process, wherein the API comprises interfaces to one or more methods of the compilation process configured for use by processes executing within the virtual machine to convert computer programming language objects into data representation language representations of the objects.

9. (Original) The method as recited in claim 1, wherein said data representation language is eXtensible Markup Language (XML).

10. (Original) The method as recited in claim 1, wherein said computer programming language is the Java programming language.

11. (Original) The method as recited in claim 1, wherein the virtual machine is a Java Virtual Machine (JVM).

12. (Original) A method for generating computer programming language objects from data representation language representations of the objects, the method comprising:

a virtual machine receiving a data representation language representation of a first computer programming language object from a first process;

a decompilation process of the virtual machine generating the first object from the data representation language representation of the first object, wherein the first object is an instance of a class in the computer programming language; and

the decompilation process of the virtual machine providing the first object to a second process executing within the virtual machine.

13. (Original) The method as recited in claim 12, wherein the first object references one or more computer programming language objects, wherein the representation of the first object includes representations of the one or more referenced objects.

14. (Original) The method as recited in claim 13, wherein the decompilation process of the virtual machine generating the first object from the representation of the first object comprises the decompilation process generating the one or more referenced objects from the representations of the one or more referenced objects included in the representation of the first object.

15. (Original) The method as recited in claim 12, wherein the decompilation process generating the first object from the data representation language representation of the first object comprises:

processing the data representation language representation of the first object into an intermediary table representation of the first object; and

generating the first object from the intermediary table representation of the first object.

16. (Original) The method as recited in claim 15, wherein the data representation language representation of the first object comprises one or more elements each representing an instance variable of the first object, wherein each element in the data representation language representation comprises an identifier for the instance variable represented by the element and a value for the instance variable represented by the element.

17. (Original) The method as recited in claim 16, wherein said processing the data representation language representation of the first object into an intermediary table representation of the first object comprises generating an entry in the intermediary table representation of the first object for each of the one or more elements in the data representation language representation of the first object.

18. (Original) The method as recited in claim 17, wherein said generating the first object from the intermediary table representation of the first object comprises:

instantiating the first object as an instance of the class; and

for each of the one or more entries in the intermediary table representation of the first object, initializing a corresponding instance variable in the first object in accordance with the entry.

19. (Original) The method as recited in claim 17, wherein said processing the intermediary table representation of the first object into the first object comprises:

instantiating the first object as an instance of the class; and

for each of the one or more entries in the intermediary table representation of the first object, invoking a method corresponding to the identifier of the instance variable from the entry to initialize a corresponding instance variable in the first object to the value of the instance variable from the entry.

20. (Original) The method as recited in claim 12, wherein the data representation language representation of the first object comprises an identifier of the class of the first object, and wherein the decompilation process generating the first object from the data representation language representation of the first object comprises instantiating the first object as an instance of the class associated with the class identifier.

21. (Original) The method as recited in claim 12, further comprising providing an application programming interface (API) for the decompilation process, wherein the API comprises interfaces to one or more methods of the decompilation process configured for use by processes executing within the virtual machine to generate computer programming language objects from data representation language representations of the objects.

22. (Original) The method as recited in claim 12, wherein said data representation language is eXtensible Markup Language (XML).

23. (Original) The method as recited in claim 12, wherein said computer programming language is the Java programming language.

24. (Original) The method as recited in claim 12, wherein the virtual machine is a Java Virtual Machine (JVM).

25. (Original) A method for passing computer programming language objects between processes in a distributed computing environment, comprising:

a first virtual machine receiving from a first process a computer programming language object, wherein the object is an instance of a class in the computer programming language;

the first virtual machine generating a representation of the object in a data representation language subsequent to said receiving;

generating a message in the data representation language, wherein the message includes the data representation language representation of the object;

sending the message to a second process; and

the second process generating a copy of the computer programming language object from the data representation language representation of the object included in the message.

26. (Original) The method as recited in claim 25, wherein the object references one or more computer programming language objects, and wherein said generating a representation of the object in a data representation language comprises generating data representation language representations of the one or more objects.

27. (Original) The method as recited in claim 25, wherein the object comprises one or more instance variables, and wherein said generating a representation of the object in a data representation language comprises:

for each of the one or more instance variables in the object, generating an element in the data representation language representation of the first object, wherein the element for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable.

28. (Original) The method as recited in claim 25, wherein the second process generating the copy of the object comprises:

the second process receiving the message including the data representation language representation of the object;

the second process providing the data representation language representation of the object to a second virtual machine;

the second virtual machine generating the copy of the object from the data representation language representation of the object; and

the second virtual machine providing the copy of the object to the second process.

29. (Original) The method as recited in claim 28, wherein the first object references one or more computer programming language objects, wherein the data representation language representation of the first object includes data representation language representations of the one or more referenced objects, and wherein said generating the copy of the object from the data representation language representation of the object comprises generating copies of the one or more referenced objects from the data representation language representations of the one or more referenced objects.

30. (Original) The method as recited in claim 28, wherein the data representation language representation of the object comprises one or more elements each representing an instance variable of the object, and wherein said generating the copy of the object from the data representation language representation of the object comprises:

instantiating the copy of the object as an instance of the class; and

for each of the one or more elements in the data representation language representation of the object, initializing a corresponding instance variable in the copy of the object in accordance with the element.

31. (Original) The method as recited in claim 25, wherein said data representation language is eXtensible Markup Language (XML).

32. (Original) The method as recited in claim 25, wherein said computer programming language is the Java programming language.

33. (Original) The method as recited in claim 25, wherein the first virtual machine is a Java Virtual Machine (JVM).

34. (Original) A method for passing computer programming language objects between processes in a distributed computing environment, comprising:

a first process receiving a message in a data representation language from a second process, wherein the message includes information representing a computer programming language object;

the first process providing the information representing the object to a virtual machine;

the virtual machine generating the object from the information representing the object, wherein the object is an instance of a class in the computer programming language; and

the virtual machine providing the generated object to the first process.

35. (Original) The method as recited in claim 34, wherein the information representing the object comprises information representing one or more instance variables of the object, wherein the information representing each of the one or more instance variables comprises an identifier for the instance variable and a value for the instance variable.

36. (Original) The method as recited in claim 35, wherein said generating the object from the information representing the object comprises:

instantiating the object as an instance of the class; and

for each of the one or more instance variables, initializing a corresponding instance variable in the object in accordance with the information representing the instance variable.

37. (Original) The method as recited in claim 34, wherein said data representation language is eXtensible Markup Language (XML).

38. (Original) The method as recited in claim 34, wherein said computer programming language is the Java programming language.

39. (Original) The method as recited in claim 34, wherein the virtual machine is a Java Virtual Machine (JVM).

40. (Original) A device comprising:

a processor;

a memory comprising virtual machine program instructions executable on the processor to:

convert a first computer programming language object into a data representation language representation of the first object, wherein the first object is an instance of a class in the computer programming language;

wherein the data representation language representation of the first object is configured for use in generating a copy of the first object.

41. (Original) The device as recited in claim 40, wherein the virtual machine program instructions are further executable to:

provide a virtual machine; and

receive the first computer programming language object from a process executing within the virtual machine.

42. (Original) The device as recited in claim 40, wherein, in said converting the first object into a data representation language representation of the first object, the virtual machine program instructions are further executable to:

process the first object into an intermediary table representation of the first object; and

process the intermediary table representation of the first object into the data representation language representation of the first object.

43. (Original) The device as recited in claim 42, wherein the first object comprises one or more instance variables, and wherein, in said processing the first object into an intermediary table representation, the virtual machine program instructions are further executable to:

generate an entry in the intermediary table representation of the first object for each of the one or more instance variables in the first object, wherein the entry for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable.

44. (Original) The device as recited in claim 43, wherein, in said processing the intermediary table representation of the first object into the data representation language representation of the first object, the virtual machine program instructions are further executable to:

generate a corresponding element in the data representation language representation of the first object for each of one or more entries in the intermediary table representation of the first object, wherein the element in

the data representation language representation of the first object includes an identifier of the instance variable and a value of the instance variable;

wherein the one or more elements in the data representation language representation of the first object are configured for use in initializing one or more corresponding instance variables in the copy of the first object.

45. (Original) The device as recited in claim 44, wherein the memory further comprises:

an application programming interface (API) to the virtual machine program instructions, wherein the API is configured to:

receive as input computer programming language objects for conversion into data representation language representations of the objects; and

provide as output the data representation language representations of the objects.

46. (Original) The device as recited in claim 40, wherein said data representation language is eXtensible Markup Language (XML).

47. (Original) The device as recited in claim 40, wherein said computer programming language is the Java programming language.

48. (Original) The device as recited in claim 40, wherein the virtual machine program instructions are further executable to provide a virtual machine configured to process bytecode into program instructions executable on the processor.

49. (Original) The device as recited in claim 48, wherein the virtual machine is a Java Virtual Machine (JVM).

50. (Original) A device, comprising:

a processor;

a memory comprising virtual machine program instructions executable on the processor to:

receive a data representation language representation of a first computer programming language object;

generate the first object from the data representation language representation of the first object, wherein the first object is an instance of a class in the computer programming language;

51. (Original) The device as recited in claim 50, wherein the virtual machine program instructions are further executable to:

provide a virtual machine; and

provide the generated first object to a process executing on the virtual machine.

52. (Original) The device as recited in claim 50, wherein the first object references one or more computer programming language objects, wherein the representation of the first object includes representations of the one or more referenced objects, wherein, in said generating the first object from the representation of the first object, the virtual machine program instructions are further executable to generate the one or more

referenced objects from the representations of the one or more referenced objects included in the representation of the first object.

53. (Original) The device as recited in claim 50, wherein, in said generating the first object from the data representation language representation of the first object, the virtual machine program instructions are further executable to:

process the data representation language representation of the first object into an intermediary table representation of the first object; and

generate the first object from the intermediary table representation of the first object.

54. (Original) The device as recited in claim 50, wherein the data representation language representation of the first object comprises one or more elements each representing an instance variable of the first object, wherein each element in the data representation language representation comprises an identifier for the instance variable represented by the element and a value for the instance variable represented by the element.

55. (Original) The device as recited in claim 54, wherein, in said processing the data representation language representation of the first object into an intermediary table representation of the first object, the virtual machine program instructions are further executable to generate an entry in the intermediary table representation of the first object for each of the one or more elements in the data representation language representation of the first object.

56. (Original) The device as recited in claim 55, wherein, in said processing the intermediary table representation of the first object into the first object, the virtual machine program instructions are further executable to:

instantiate the first object as an instance of the class; and

for each of the one or more entries in the intermediary table representation of the first object, invoke a method corresponding to the identifier of the instance variable from the entry, wherein the method is operable to initialize a corresponding instance variable in the first object to the value of the instance variable from the entry.

57. (Original) The device as recited in claim 51, further comprising:

an application programming interface (API) to the virtual machine program instructions, wherein the API is configured to:

receive as input the data representation language representation of the first object; and

provide as output the generated first object.

58. (Original) The device as recited in claim 50, wherein said data representation language is eXtensible Markup Language (XML).

59. (Original) The device as recited in claim 50, wherein said computer programming language is the Java programming language.

60. (Original) The device as recited in claim 50, wherein the virtual machine program instructions are further executable to provide a virtual machine configured to process bytecode into program instructions executable on the processor.

61. (Original) The device as recited in claim 60, wherein the virtual machine is a Java Virtual Machine (JVM).

62. (Original) A distributed computing system, comprising:

a first device comprising:

a first processor;

a first memory comprising:

virtual machine program instructions executable on the first processor to provide a first virtual machine; and

a first process executable within the first virtual machine;

wherein the first virtual machine is operable to:

receive from the first process a computer programming language object, wherein the object is an instance of a class in the computer programming language; and

generate a representation of the object in a data representation language subsequent to said receiving;

wherein the first process is operable to:

generate a message in the data representation language, wherein the message includes the data representation language representation of the object.

63. (Original) The system as recited in claim 62, further comprising:

a second device comprising:

a second processor;

a second memory comprising:

virtual machine program instructions executable on the second processor to provide a second virtual machine;

a second process executable within the second virtual machine;

wherein the first process is further operable to send the message to the second process.

64. (Original) The system as recited in claim 63,

wherein the second process is operable to:

receive the message including the data representation language representation of the object; and

provide the data representation language representation of the object to the second virtual machine;

wherein the second virtual machine is operable to:

generate a copy of the object from the data representation language representation of the object; and

provide the copy of the object to the second process.

65. (Original) The system as recited in claim 63, wherein the first virtual machine and the second virtual machine are further operable to process bytecode into program instructions executable on the processor.

66. (Original) The system as recited in claim 63, wherein the first virtual machine and the second virtual machine are Java Virtual Machines (JVMs).

67. (Original) The system as recited in claim 62, wherein the first object references one or more computer programming language objects, wherein the data representation language representation of the first object includes data representation language representations of the one or more referenced objects, and wherein, in said generating the copy of the object from the data representation language representation of the object, the second virtual machine is further operable to generate copies of the one or more referenced objects from the data representation language representations of the one or more referenced objects.

68. (Original) The system as recited in claim 62, wherein the data representation language representation of the object comprises one or more elements each representing an instance variable of the object, and wherein, in said generating the copy of the object from the data representation language representation of the object, the second virtual machine is further operable to:

instantiate the copy of the object as an instance of the class; and

for each of the one or more elements in the data representation language representation of the object, initialize a corresponding instance variable in the copy of the object in accordance with the element.

69. (Original) The system as recited in claim 62, wherein said data representation language is eXtensible Markup Language (XML).

70. (Original) The system as recited in claim 62, wherein said computer programming language is the Java programming language.

71. (Currently amended) A ~~carrier~~ computer accessible medium comprising program instructions, wherein the program instructions are computer-executable to implement:

a process executing within a virtual machine providing a first computer programming language object to a compilation process of the virtual machine, wherein the first object is an instance of a class in the computer programming language; and

the compilation process of the virtual machine converting the first object into a data representation language representation of the first object;

wherein the data representation language representation of the first object is configured for use in generating a copy of the first object.

72. (Currently amended) The ~~system~~ computer accessible medium as recited in claim 71, wherein, in said converting the first object into a data representation language representation of the first object, the program instructions are further computer-executable to implement:

processing the first object into an intermediary table representation of the first object; and

processing the intermediary table representation of the first object into the data representation language representation of the first object.

73. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 72, wherein the first object comprises one or more instance variables, and wherein, in

said processing the first object into an intermediary table representation, the program instructions are further computer-executable to implement:

for each of the one or more instance variables in the first object, generating an entry in the intermediary table representation of the first object, wherein the entry for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable.

74. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 73, wherein, in said processing the intermediary table representation of the first object into the data representation language representation of the first object, the program instructions are further computer-executable to implement:

for each of one or more entries in the intermediary table representation of the first object, generating a corresponding element in the data representation language representation of the first object, wherein the element in the data representation language representation of the first object includes an identifier of the instance variable and a value of the instance variable.

75. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 71, wherein said data representation language is eXtensible Markup Language (XML).

76. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 71, wherein said computer programming language is the Java programming language.

77. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 71, wherein the virtual machine is a Java Virtual Machine (JVM).

78. (Currently amended) A ~~carrier~~ computer accessible medium comprising program instructions, wherein the program instructions are computer-executable to implement:

a virtual machine receiving a data representation language representation of a first computer programming language object from a first process;

a decompilation process of the virtual machine generating the first object from the data representation language representation of the first object, wherein the first object is an instance of a class in the computer programming language; and

the decompilation process of the virtual machine providing the first object to a second process executing within the virtual machine.

79. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 78, wherein, in said generating the first object from the data representation language representation of the first object, the program instructions are further computer-executable to implement:

processing the data representation language representation of the first object into an intermediary table representation of the first object; and

generating the first object from the intermediary table representation of the first object.

80. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 79, wherein, in said generating the first object from the intermediary table representation of the first object, the program instructions are further computer-executable to implement:

instantiating the first object as an instance of the class; and

for each of the one or more entries in the intermediary table representation of the first object, initializing a corresponding instance variable in the first object in accordance with the entry.

81. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 78, wherein said data representation language is eXtensible Markup Language (XML).

82. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 78, wherein said computer programming language is the Java programming language.

83. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 78, wherein the virtual machine is a Java Virtual Machine (JVM).

84. (Currently amended) A ~~carrier~~ computer accessible medium comprising program instructions, wherein the program instructions are computer-executable to implement:

a first virtual machine receiving from a first process a computer programming language object, wherein the object is an instance of a class in the computer programming language;

the first virtual machine generating a representation of the object in a data representation language subsequent to said receiving;

generating a message in the data representation language, wherein the message includes the data representation language representation of the object;

sending the message to a second process; and

the second process generating a copy of the computer programming language object from the data representation language representation of the object included in the message.

85. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 84, wherein the object comprises one or more instance variables, and wherein, in said generating a representation of the object in a data representation language, the program instructions are further computer-executable to implement:

for each of the one or more instance variables in the object, generating an element in the data representation language representation of the first object, wherein the element for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable.

86. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 84, wherein, in said generating the copy of the object, the program instructions are further computer-executable to implement:

the second process receiving the message including the data representation language representation of the object;

the second process providing the data representation language representation of the object to a second virtual machine;

the second virtual machine generating the copy of the object from the data representation language representation of the object; and

the second virtual machine providing the copy of the object to the second process.

87. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 86, wherein the data representation language representation of the object comprises one or more elements each representing an instance variable of the object, and wherein, in said generating the copy of the object from the data representation language representation of the object, the program instructions are further computer-executable to implement:

instantiating the copy of the object as an instance of the class; and

for each of the one or more elements in the data representation language representation of the object, initializing a corresponding instance variable in the copy of the object in accordance with the element.

88. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 84, wherein said data representation language is eXtensible Markup Language (XML).

89. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 84, wherein said computer programming language is the Java programming language.

90. (Currently amended) The ~~carrier~~ computer accessible medium as recited in claim 84, wherein the first virtual machine is a Java Virtual Machine (JVM).